Algorithms in Bioinformatics
Problem set 1

Programming assignments

P1: Graph Traversal Algorithms

a) Write a function that takes the name of a .csv file as input, reads the list of undirected edges from the file, and creates an adjacency matrix from the interactions. Assume that the input is a simple graph. See this file for an Example:

b) Write a function that, given the adjacency matrix of a graph, performs DFS and returns the resulting tree as an adjacency matrix.

c) Write a function that, given the adjacency matrix of a graph, performs BFS and returns the resulting tree as an adjacency matrix.

a)

```
> ###P1: Graph Traversal Algorithms
> ####a
> ###my_net= read.csv("cancer_healthy_countdata.csv")
> net_1= read_excel("edge_list.xlsx")
> net_1$node1
 [1] "a" "a" "a" "b" "c" "c" "d" "f" "f" "g" "b" "e"
> net_1$node2
 [1] "b" "c" "d" "d" "e" "d" "f" "g" "h" "h" "e" "h"
>  nodes= unique(c(net_1$node1 , net_1$node2))
>  adj_mat= matrix(0 , nrow = length(nodes) , ncol = length(nodes))
> rownames(adj_mat)= nodes
> colnames(adj_mat) = nodes
> for (i in 1: nrow(net_1)) {
+   u= net_1[i , 1]
+   v= net_1[i , 2]
+   adj_mat[as.character(u) ,as.character(v)]= 1
+
+ }
> adj_mat
  a b c d f g e h
a 0 1 1 1 0 0 0 0
b 0 0 0 1 0 0 1 0
c 0 0 0 1 0 0 1 0
d 0 0 0 0 1 0 0 0
f 0 0 0 0 0 1 0 1
g 0 0 0 0 0 0 0 1
e 0 0 0 0 0 0 0 1
h 0 0 0 0 0 0 0 0
```

b)

```
> dfs = function(adj_mat1) {
+   # Initialize the stack and visited set
+   ###nodes1=nodes1=c("a" , "b" , "c" , "d")
+   stack =c(1)
+   visited = c()
+   dfs_tree = matrix(0, ncol = ncol(adj_mat1), nrow = nrow(adj_mat1))
+   # Loop until the stack is empty
+   while (length(stack) > 0) {
+     # Pop the top node from the stack
+     current = stack[length(stack)]
+     stack = stack[-length(stack)]
+
+     # Visit the node if it hasn't been visited before
+     if (!(current %in% visited)) {
+       visited = c(visited, current)
+
+
+       # Add the unvisited neighbors to the stack
+       neighbors = which(adj_mat1[current,] != 0)
+       for (neighbors in neighbors) {
+         unvisited = neighbors[!(neighbors %in% visited)]
+         stack = c(stack, unvisited)
+         dfs_tree[current, neighbors] = 1
+       }
+
+     }
+   }
+   dfs_tree
+ }
```

```
> adj_mat1 <- matrix(c(0,1,1,0,0,
+                      1,0,1,1,0,
+                      1,1,0,1,1,
+                      0,1,1,0,1,
+                      0,0,1,1,0), nrow=5, ncol=5, byrow=TRUE)
> dfs(adj_mat1)
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    1    1    0    0
[2,]    1    0    1    1    0
[3,]    1    1    0    1    1
[4,]    0    1    1    0    1
[5,]    0    0    1    1    0
```

c)

```
> bfs <- function(adj_matrix) {
+   # Initialize the queue and visited set
+   queue <- c(1)
+   visited <- c()
+   n_nodes <- nrow(adj_matrix)
+   tree = matrix(0, nrow=n_nodes, ncol=n_nodes)
+
+   # Loop until the queue is empty
+   while (length(queue) > 0) {
+     # Dequeue the front node from the queue
+     current_node <- queue[1]
+     queue <- queue[-1]
+
+     # Visit the node if it hasn't been visited before
+     if (!(current_node %in% visited)) {
+       visited <- c(visited, current_node)
+
+
+       # Add the unvisited neighbors to the queue
+       neighbors <- which(adj_matrix[current_node,] != 0)
+       unvisited_neighbors <- neighbors[!(neighbors %in% visited)]
+       queue <- c(queue, unvisited_neighbors)
+       tree[current_node, unvisited_neighbors] = 1
+       print(visited)
+
+     }
+   }
+
+   return(tree)
+
+ }

> # Define the adjacency matrix of the graph
> adj_matrix <- matrix(c(0,1,1,0,0,
+                        1,0,1,1,0,
+                        1,1,0,1,1,
+                        0,1,1,0,1,
+                        0,0,1,1,0), nrow=5, ncol=5, byrow=TRUE)
> # Call the bfs function with starting node 1
> bfs_tree <- bfs(adj_matrix)
[1] 1
[1] 1 2
[1] 1 2 3
[1] 1 2 3 4
[1] 1 2 3 4 5
> # Print the order of visited nodes
> print(bfs_tree)
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    1    1    0    0
[2,]    0    0    1    1    0
[3,]    0    0    0    1    1
[4,]    0    0    0    0    1
[5,]    0    0    0    0    0
```

P3: A gene expression data is simulated for ten healthy participants (rows 1 to 10) and ten cancer patients (rows 11 to 20) and on 12500 genes. The goal is to find significantly different genes between the two groups. In the simulated gene expression matrix, only the first 1250 genes are different between two groups. Develop three various statistics (as we defined in the lecture) and compute empirical p-values. Compare your approaches with the t-test and Wilcoxon test in terms of type I and type II errors and other criteria of your choice. Perform a comprehensive analysis.

Develop three various statistics and compute empirical p-values:

1: resampling methods : mean

```
####mean
data_P3= read.csv("Cancer_healthy_countdata.csv")
set.seed(123)
#### Initialize a vector to store the p-values for each gene
pvalue_all = c()
# For each gene, calculate the mean expression level for each group
####3000 genes
for (i in 2 :3000) {
  # For each gene, calculate the mean expression level for each group and
  ##the mean difference between the two groups
  differ_mean= mean(data_P3[1:10 , i]) - mean(data_P3[11:20 , i])
  ###for each gene of k samples
  sample_diff= c()
  ###### Define the number of iterations for resampling  1000
  for (k in 1:1000) {

    ####  # Use resampling to compute an empirical p-value for the gene

    sample_index= sample(nrow(data_P3) , 10)
    sample_differ_k= mean(data_P3[sample_index , i])-
      mean(data_P3[-sample_index , i])

    if(abs(sample_differ_k) < abs(differ_mean)){
      sample_diff= c(sample_diff , sample_differ_k)
    }
    pvalu_gene= length(sample_diff)/ 1000

  }
  pvalue_all = c(pvalue_all , pvalu_gene)

}
pvalue_all
# Use a significance threshold of 0.05 to identify significantly different genes
significant_genes = which(pvalue_all < 0.05)
```

```
> significant_genes = which(pvalue_all < 0.05)
> significant_genes
  [1]  302  307  392  710  732  734  864 1285 1294 1299 1316 1326 1350 1365 1373 1394 1444
 [18] 1449 1472 1501 1523 1524 1568 1572 1574 1591 1600 1612 1638 1646 1661 1662 1674 1698
 [35] 1715 1733 1744 1757 1790 1862 1870 1881 1882 1883 1900 1907 1952 1958 2002 2012 2017
 [52] 2021 2032 2061 2082 2115 2146 2150 2155 2163 2173 2189 2193 2205 2206 2214 2251 2258
 [69] 2263 2266 2285 2298 2331 2369 2374 2388 2396 2397 2402 2404 2433 2440 2465 2500 2518
 [86] 2522 2542 2543 2544 2550 2553 2573 2633 2657 2659 2668 2690 2694 2703 2715 2717 2762
[103] 2767 2809 2813 2825 2829 2875 2883 2937 2975 2988 2998
```

در این مسئله مشخص شده است که ۱۲۵۰ تا ژن اول **DEG** هستند و برای ژن های بعد از ۱۲۵۰ فرض اولیه بر این است که اختلاف میانگین نداریم. حال **H null** را به این صورت زیر در نظر گرفته شده است که اختلاف میانگین بین دو گروه وجود ندارد

Genes: 1 : 3000

$H0 : \mu1=\mu2$ and $H1: \mu1-\mu2!= 0$

p-value< 0.05 reject H0

p-value> 0.05 dont reject H0

```
> pvalue_all
   [1] 0.434 1.000 1.000 0.970 1.000 1.000 0.962 1.000 0.901 1.000 1.000 0.954 0.994 1.000
  [15] 1.000 1.000 0.998 0.882 0.882 0.620 0.993 0.107 0.287 0.425 0.991 1.000 0.989 1.000
  [29] 0.999 0.742 0.992 0.993 1.000 1.000 0.403 0.999 0.998 0.999 1.000 1.000 1.000 1.000
  [43] 1.000 0.998 0.877 1.000 1.000 0.985 1.000 0.994 0.405 0.997 1.000 1.000 1.000 0.292
  [57] 0.942 1.000 0.997 1.000 1.000 0.285 0.999 0.931 0.971 0.997 0.999 1.000 0.964 0.998
  [71] 0.995 1.000 0.918 0.992 0.999 1.000 0.942 0.865 1.000 0.710 0.737 1.000 1.000 0.999
  [85] 0.434 0.999 1.000 0.802 1.000 0.998 1.000 1.000 0.994 0.994 0.981 1.000 0.530 0.507
  [99] 0.708 1.000 0.988 0.983 0.999 0.998 1.000 1.000 0.940 0.993 0.096 0.930 1.000 1.000
 [113] 0.874 0.866 0.999 0.999 1.000 1.000 1.000 0.996 0.443 1.000 0.933 1.000 0.999 0.997
 [127] 0.971 0.824 0.988 0.563 1.000 0.999 1.000 0.915 0.997 0.964 0.969 0.587 1.000 1.000
 [141] 0.998 0.998 1.000 1.000 1.000 0.992 0.998 1.000 1.000 0.984 1.000 1.000 0.993 0.068
 [155] 1.000 1.000 0.589 0.886 1.000 0.972 0.910 1.000 1.000 1.000 0.136 1.000 0.842 0.962
 [169] 0.271 0.997 0.993 1.000 0.963 1.000 1.000 0.795 1.000 0.996 0.977 0.935 1.000 1.000
 [183] 1.000 0.650 1.000 0.857 0.997 0.989 1.000 0.897 1.000 0.775 1.000 1.000 0.940 0.576
 [197] 0.995 0.969 0.802 0.997 0.889 0.997 0.623 1.000 1.000 0.994 0.999 0.968 1.000 1.000
 [211] 0.999 0.993 0.357 1.000 0.992 0.865 1.000 1.000 0.998 0.757 0.993 1.000 1.000 1.000
 [225] 0.126 1.000 0.932 1.000 0.671 0.995 0.989 0.998 0.253 1.000 0.845 0.999 0.516 0.958
 [239] 1.000 0.942 0.898 0.994 1.000 0.936 1.000 0.989 0.926 0.251 1.000 0.995 0.990 0.960
 [253] 0.848 1.000 0.321 1.000 1.000 1.000 0.950 0.866 0.975 0.943 0.998 0.998 0.800 1.000
 [267] 0.995 0.634 0.994 1.000 0.994 0.995 1.000 0.982 0.997 0.989 0.942 1.000 0.946 1.000
 [281] 1.000 0.918 1.000 0.989 0.427 0.999 1.000 1.000 1.000 0.998 1.000 0.852 1.000 1.000
 [295] 0.970 0.935 0.667 0.486 0.418 0.262 0.998 0.011 0.571 0.169 0.996 0.646 0.000 0.790
 [309] 0.723 1.000 1.000 0.998 1.000 0.999 0.900 1.000 0.999 0.999 0.952 0.997 0.898 1.000
 [323] 0.970 1.000 0.989 0.998 0.828 1.000 0.997 1.000 1.000 0.999 1.000 0.997 1.000 0.605
 [337] 0.857 1.000 0.995 0.275 1.000 0.522 1.000 1.000 0.993 0.972 0.971 1.000 1.000 0.964
 [351] 1.000 0.991 1.000 0.438 0.997 1.000 0.263 0.953 1.000 0.883 0.971 0.999 0.479 0.950
 [365] 1.000 0.865 1.000 1.000 0.979 0.997 0.983 1.000 1.000 1.000 1.000 0.996 0.898 0.990
 [379] 1.000 1.000 1.000 0.979 1.000 1.000 1.000 0.760 0.711 0.986 0.899 0.988 0.674 0.029
 [393] 1.000 1.000 0.980 0.993 0.998 0.995 0.779 1.000 0.912 0.998 0.941 0.996 0.991 0.527
 [407] 1.000 0.998 0.523 1.000 0.466 0.999 1.000 1.000 0.907 0.999 0.994 1.000 0.990 0.998
 [421] 1.000 1.000 1.000 1.000 1.000 1.000 1.000 0.999 1.000 0.997 1.000 0.998 1.000 0.999
 [435] 1.000 0.874 0.973 1.000 0.994 1.000 0.628 0.996 0.936 0.981 1.000 0.828 0.972 0.384
 [449] 0.928 1.000 0.999 0.941 1.000 0.814 0.990 0.987 0.999 0.998 1.000 1.000 0.948 1.000
 [463] 1.000 0.947 0.997 0.999 1.000 0.838 0.996 0.996 0.772 1.000 0.918 0.811 0.999 1.000
 [477] 1.000 1.000 0.719 0.995 1.000 0.825 0.994 0.889 0.998 0.999 0.771 0.999 0.993 0.975
 [491] 0.924 0.978 0.997 0.994 1.000 1.000 0.999 1.000 0.996 1.000 0.880 0.995 0.974 0.629
```

Type I and II Errors:

Decision	H_0 True	H_0 False
Do Not Reject H_0	Correct Decision $1 - \alpha$	Incorrect Decision Type II Error β
Rejct H_0	Incorrect Decision Type I Error α	Correct Decision $1 - \beta$

Actual Situation "Truth"

$$\alpha = P(Type\ I\ Error) \quad \beta = P(Type\ II\ Error)$$

در این قسمت از ژن های در بازه ۱:۱۲۵۰ که **DEG** هستند و اختلاف میانگین برای هر ژن در سمپل های سالم و کنسر به صورت معنادار هست پس **H null** را می توان ریجکت کرد. اما همانطور که

در نتیجه زیر مشاهده می شود ژن های ۱: ۱۲۵۰ مواردی را داریم که **p.value**> ۰,۰۵ هست پس اینجا خطا رخ داده است.

```
> significant_genes = which(pvalue_all < 0.05)
> significant_genes
  [1]  302  307  392  710  732  734  864 1285 1294 1299 1316 1326 1350 1365 1373 1394 1444
 [18] 1449 1472 1501 1523 1524 1568 1572 1574 1591 1600 1612 1638 1646 1661 1662 1674 1698
 [35] 1715 1733 1744 1757 1790 1862 1870 1881 1882 1883 1900 1907 1952 1958 2002 2012 2017
 [52] 2021 2032 2061 2082 2115 2146 2150 2155 2163 2173 2189 2193 2205 2206 2214 2251 2258
 [69] 2263 2266 2285 2298 2331 2369 2374 2388 2396 2397 2402 2404 2433 2440 2465 2500 2518
 [86] 2522 2542 2543 2544 2550 2553 2573 2633 2657 2659 2668 2690 2694 2703 2715 2717 2762
[103] 2767 2809 2813 2825 2829 2875 2883 2937 2975 2988 2998
```

در پوزیشن **i=** ۳۰۲ به طور مثال می توان گفت فرض اولیه که برابری میانگین ها را در نطر گرفته که اشتباه هست و **p.value** هم کمتر از ۰,۰۵ شده است و پس فرض اولیه ریجکت می شود در این قسمت می توان گفت

1-beta : correct dicision

در پوزیشن هایی که در این بازه هستند و **pvalue**> ۰,۰۵ باشد با فرض اولیه اشتباه و عدم ریجکت آن خطای نوع دوم **beta** داریم .

برای بازه ۱۲۵۰ تا ۳۰۰۰ تا ژنی که در نظر گرفتیم با فرض برابری میانگین ها و **p.value** هایی که کمتر از ۰,۰۵ هستند و با ریجکت فرض اول خطای نوع اول داریم **alpha**. برای تک تک ژن ها که دارای **pvalue** هستند خطاها را می توان حساب کرد.

2: resampling methods : median

```
######median
data_P3= read.csv("Cancer_healthy_countdata.csv")
set.seed(123)
#### Initialize a vector to store the p-values for each gene
pvalue_all_median = c()
# For each gene, calculate the median expression level for each group
####3000 genes
for (i in 2 :3000) {
  # For each gene, calculate the median expression level for each group and
  ##the median difference between the two groups
  differ_median= median(data_P3[1:10 , i]) - median(data_P3[11:20 , i])
  ###for each gene of k samples
  sample_diff_median= c()
  ###### Define the number of iterations for resampling  1000
  for (k in 1:1000) {

    ####  # Use resampling to compute an empirical p-value for the gene

    sample_index= sample(nrow(data_P3) , 10)
    sample_differ_median_k= median(data_P3[sample_index , i])-
      median(data_P3[-sample_index , i])

    if(abs(sample_differ_median_k) < abs(differ_median)){
      sample_diff_median= c(sample_diff_median , sample_differ_median_k)
    }
    pvalu_gene_median= length(sample_diff_median)/ 1000

  }
  pvalue_all_median = c(pvalue_all_median , pvalu_gene_median)

}
pvalue_all_median
# Use a significance threshold of 0.05 to identify significantly different genes
significant_genes_median = which(pvalue_all_median < 0.05)
```

```
> pvalue_all_median
   [1] 0.000 0.998 1.000 0.899 0.991 0.999 0.981 0.998 0.810 0.997 1.000 0.882 0.989 1.000
  [15] 1.000 0.998 0.990 0.723 0.318 0.161 0.992 0.016 0.506 0.118 0.994 1.000 0.992 0.999
  [29] 0.997 0.231 0.994 0.992 1.000 1.000 0.350 0.998 0.998 1.000 1.000 1.000 0.997 0.999
  [43] 1.000 0.992 0.734 0.999 0.997 0.990 1.000 0.995 0.435 0.998 1.000 0.999 1.000 0.047
  [57] 0.952 0.999 0.994 0.999 0.999 0.063 0.986 0.915 0.935 0.988 0.999 1.000 0.896 0.995
  [71] 0.984 0.995 0.892 0.982 0.999 0.998 0.862 0.686 0.997 0.754 0.416 1.000 0.989 0.976
  [85] 0.002 0.995 1.000 0.748 0.999 0.999 0.999 0.994 0.995 0.989 0.953 1.000 0.532 0.618
  [99] 0.507 1.000 0.990 0.979 0.998 0.993 0.998 0.998 0.624 0.998 0.838 0.763 0.999 0.999
 [113] 0.914 0.577 1.000 0.999 0.999 0.999 0.997 0.986 0.525 0.999 0.888 0.999 0.999 0.988
 [127] 0.963 0.498 0.943 0.000 1.000 0.995 0.999 0.792 0.995 0.943 0.979 0.674 1.000 1.000
 [141] 0.987 0.999 1.000 1.000 0.999 0.987 0.996 0.998 1.000 0.982 0.999 0.996 0.994 0.000
 [155] 0.998 1.000 0.845 0.424 1.000 0.886 0.930 0.998 0.999 0.999 0.475 1.000 0.824 0.913
 [169] 0.170 0.989 0.996 1.000 0.906 1.000 0.999 0.871 1.000 0.988 0.990 0.970 1.000 0.999
 [183] 1.000 0.000 0.999 0.448 0.999 0.971 0.999 0.885 0.999 0.555 1.000 0.999 0.913 0.808
 [197] 0.979 0.946 0.948 0.998 0.919 0.980 0.945 0.999 1.000 0.996 0.998 0.943 0.999 0.998
 [211] 0.998 0.965 0.420 1.000 0.992 0.960 0.997 0.999 0.999 0.839 0.997 1.000 0.999 0.995
 [225] 0.043 1.000 0.932 0.998 0.851 0.944 0.989 0.999 0.490 0.999 0.650 1.000 0.255 0.888
 [239] 1.000 0.921 0.856 0.998 1.000 0.905 1.000 0.956 0.925 0.036 0.999 0.997 0.990 0.948
 [253] 0.348 0.999 0.398 0.999 0.999 0.999 0.972 0.271 0.953 0.976 0.998 0.998 0.185 0.998
 [267] 0.972 0.753 0.996 1.000 0.961 0.994 0.999 0.990 0.989 0.993 0.945 0.997 0.901 1.000
 [281] 1.000 0.600 1.000 0.975 0.000 0.996 0.998 1.000 0.999 0.973 1.000 0.961 0.999 0.997
 [295] 0.982 0.916 0.187 0.631 0.053 0.507 0.995 0.203 0.499 0.196 0.999 0.306 0.337 0.574
 [309] 0.470 0.998 0.999 0.996 0.997 0.997 0.917 1.000 0.999 0.987 0.984 0.994 0.759 1.000
 [323] 0.984 1.000 0.969 0.995 0.679 1.000 0.999 0.997 1.000 0.996 0.998 0.998 0.999 0.533
 [337] 0.826 0.999 0.951 0.141 1.000 0.662 1.000 0.995 0.990 0.904 0.935 1.000 0.999 0.960
 [351] 0.982 0.990 0.999 0.000 0.994 1.000 0.823 0.914 0.998 0.847 0.916 0.999 0.572 0.680
 [365] 1.000 0.495 1.000 0.999 0.797 0.996 0.955 0.998 1.000 0.999 0.998 0.998 0.858 0.985
 [379] 0.999 0.999 1.000 0.987 0.999 1.000 0.999 0.881 0.851 0.988 0.814 0.982 0.405 0.681
 [393] 1.000 1.000 0.970 0.988 0.997 0.991 0.402 1.000 0.587 0.998 0.840 0.982 0.991 0.447
 [407] 0.998 0.967 0.115 1.000 0.345 0.999 1.000 0.997 0.891 0.997 0.981 1.000 0.992 0.999
 [421] 0.999 1.000 0.999 0.999 1.000 1.000 0.999 0.999 0.982 0.998 0.999 0.995 1.000 0.998
 [435] 0.999 0.277 0.871 0.998 0.993 1.000 0.455 0.997 0.920 0.969 0.998 0.308 0.323 0.762
 [449] 0.952 1.000 0.998 0.803 0.986 0.752 0.990 0.984 1.000 0.998 0.999 0.997 0.915 0.999
```

```
> significant_genes_median = which(pvalue_all_median < 0.05)
> significant_genes_median
  [1]    1   22   56   85  130  154  184  225  248  285  354  582  590  609  714  732  769
 [18]  798  811  829  836  858  916  930  972  984 1009 1028 1035 1062 1082 1112 1115 1124
 [35] 1127 1139 1157 1213 1257 1266 1269 1282 1288 1297 1299 1303 1305 1311 1313 1314 1317
 [52] 1351 1353 1356 1366 1373 1383 1393 1416 1420 1427 1429 1430 1436 1446 1452 1496 1501
 [69] 1505 1506 1516 1537 1539 1543 1544 1552 1560 1561 1568 1595 1613 1630 1635 1648 1657
 [86] 1663 1677 1678 1683 1706 1723 1734 1736 1746 1747 1755 1761 1768 1769 1779 1782 1786
[103] 1811 1820 1843 1850 1858 1865 1868 1897 1900 1902 1908 1909 1911 1919 1930 1935 1939
[120] 1944 1983 1999 2002 2007 2012 2024 2034 2049 2050 2057 2064 2065 2071 2079 2082 2084
[137] 2103 2105 2109 2118 2126 2137 2143 2146 2163 2164 2167 2173 2174 2175 2185 2187 2189
[154] 2202 2211 2224 2236 2240 2241 2243 2246 2251 2268 2272 2275 2293 2297 2306 2320 2336
[171] 2338 2339 2357 2369 2382 2383 2388 2393 2404 2406 2415 2417 2425 2429 2430 2433 2437
[188] 2440 2454 2457 2459 2470 2476 2481 2490 2493 2495 2512 2513 2522 2527 2539 2543 2544
[205] 2545 2546 2551 2553 2558 2559 2561 2568 2570 2616 2624 2628 2631 2635 2643 2645 2665
[222] 2673 2679 2713 2734 2735 2737 2759 2763 2799 2806 2809 2817 2838 2841 2858 2873 2875
[239] 2879 2898 2912 2913 2941 2943 2959 2969 2975 2989
```

برای قسمت **median**:

Genes: 1 : 3000

H0 : median1=median2 and H1: median1−median2!= 0

p-value< 0.05 reject H0

p-value> 0.05 dont reject H0

طبق مطالبی که در قسمت میانگین گفته شده می توان خطای اول و دوم را برای **pvalue** های بدس آمده را حل کرد.

3: resampling methods : max

```
##########max
data_P3= read.csv("Cancer_healthy_countdata.csv")
set.seed(123)
#### Initialize a vector to store the p-values for each gene
pvalue_all_max = c()
# For each gene, calculate the max of  expression level for each group
####3000 genes
for (i in 2 :3000) {
  # For each gene, calculate the max of expression level for each group and
  ##the max difference between the two groups
  differ_max= max(data_P3[1:10 , i]) - max(data_P3[11:20 , i])
  ###for each gene of k samples
  sample_diff_max= c()
  ###### Define the number of iterations for resampling  1000
  for (k in 1:1000) {

    ####  # Use resampling to compute an empirical p-value for the gene

    sample_index= sample(nrow(data_P3) , 10)
    sample_differ_k_max= max(data_P3[sample_index , i])-
      max(data_P3[-sample_index , i])

    if(abs(sample_differ_k_max) < abs(differ_max)){
      sample_diff_max= c(sample_diff_max , sample_differ_k_max)
    }
    pvalu_gene_max= length(sample_diff_max)/ 1000

  }
  pvalue_all_max = c(pvalue_all_max , pvalu_gene_max)

}
pvalue_all_max
# Use a significance threshold of 0.05 to identify significantly different genes
significant_genes_max = which(pvalue_all_max < 0.05)
```

```
> pvalue_all_max
  [1] 0.517 1.000 1.000 0.790 0.997 1.000 0.000 1.000 0.515 1.000 0.998 0.892 0.547 0.996
 [15] 1.000 1.000 0.998 0.777 0.922 0.538 0.997 0.502 0.000 0.544 0.969 1.000 0.996 1.000
 [29] 0.995 0.908 0.988 0.990 1.000 1.000 0.000 0.990 0.995 0.988 1.000 0.988 1.000 1.000
 [43] 1.000 1.000 0.813 1.000 1.000 0.968 1.000 0.997 0.561 0.987 0.994 1.000 1.000 0.000
 [57] 0.515 1.000 0.988 1.000 0.999 0.523 0.525 0.791 0.969 0.902 0.983 1.000 0.936 0.998
 [71] 0.000 0.998 0.783 0.804 1.000 1.000 0.789 0.791 1.000 0.000 0.535 0.996 0.907 0.922
 [85] 0.000 0.998 1.000 0.781 1.000 0.988 1.000 0.992 0.994 0.985 0.979 1.000 0.000 0.000
 [99] 0.790 1.000 0.965 0.957 0.998 0.775 1.000 0.999 0.782 0.974 0.544 0.794 1.000 1.000
[113] 0.000 0.898 0.998 0.998 1.000 1.000 1.000 0.778 0.000 1.000 0.930 1.000 0.967 0.519
[127] 0.532 0.897 0.904 0.000 1.000 0.999 0.989 0.522 0.974 0.907 0.961 0.000 0.993 1.000
[141] 0.970 0.966 1.000 0.994 0.999 0.955 0.988 0.999 1.000 0.967 1.000 0.965 1.000 0.000
[155] 1.000 1.000 0.000 0.793 1.000 0.911 0.000 1.000 1.000 1.000 0.000 1.000 0.526 0.516
[169] 0.000 0.996 0.986 1.000 0.966 1.000 1.000 0.000 1.000 0.927 0.990 0.969 1.000 1.000
[183] 1.000 0.519 0.989 0.781 0.966 0.921 0.998 0.000 1.000 0.900 1.000 1.000 0.911 0.000
[197] 0.907 0.970 0.000 0.996 0.508 0.790 0.000 0.997 1.000 0.967 0.998 0.917 1.000 1.000
[211] 1.000 0.922 0.000 0.997 0.992 0.000 1.000 0.999 0.557 0.000 0.996 1.000 1.000 1.000
[225] 0.000 0.990 0.907 1.000 0.000 0.914 0.982 0.998 0.508 1.000 0.521 0.990 0.000 0.792
[239] 1.000 0.796 0.000 0.988 1.000 0.922 0.995 0.907 0.526 0.000 1.000 0.960 0.963 0.963
[253] 0.784 1.000 0.000 0.995 1.000 1.000 0.526 0.928 0.983 0.991 0.998 0.995 0.542 0.997
[267] 0.990 0.550 0.998 0.999 0.919 0.996 1.000 0.996 0.995 0.997 0.522 1.000 0.969 1.000
[281] 0.999 0.965 0.993 0.997 0.552 1.000 1.000 0.992 1.000 0.921 0.998 0.000 1.000 1.000
[295] 0.986 0.550 0.542 0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.969 0.507 0.000 0.000
[309] 0.784 1.000 1.000 0.990 0.999 0.999 0.531 0.999 0.999 0.997 0.990 0.971 0.796 0.998
[323] 0.000 1.000 0.512 1.000 0.519 1.000 0.990 1.000 1.000 0.999 1.000 0.991 1.000 0.000
[337] 0.765 1.000 0.768 0.536 1.000 0.000 1.000 1.000 0.509 0.912 0.971 1.000 1.000 0.520
[351] 1.000 0.966 1.000 0.000 0.989 1.000 0.000 0.792 1.000 0.810 0.804 0.994 0.000 0.919
[365] 0.969 0.920 1.000 0.999 0.922 0.993 0.783 1.000 1.000 1.000 1.000 0.992 0.972 0.966
[379] 1.000 1.000 1.000 0.963 1.000 1.000 1.000 0.000 0.000 0.984 0.920 0.551 0.488 0.000
[393] 0.999 1.000 0.911 0.794 0.998 0.964 0.542 1.000 0.798 0.991 0.964 0.997 0.961 0.000
[407] 1.000 0.912 0.780 1.000 0.793 0.962 1.000 1.000 0.508 1.000 0.910 1.000 0.977 0.991
[421] 1.000 0.998 0.999 1.000 1.000 1.000 1.000 0.989 0.909 0.996 1.000 0.997 1.000 0.999
[435] 0.999 0.781 0.776 1.000 0.965 1.000 0.537 0.993 0.910 0.967 1.000 0.916 0.909 0.563
[449] 0.988 1.000 0.969 0.912 0.998 0.502 0.998 0.970 0.960 0.989 1.000 0.999 0.958 1.000
[463] 1.000 0.530 0.988 0.989 0.991 0.900 0.990 0.994 0.519 1.000 0.762 0.773 0.998 0.997
[477] 1.000 1.000 0.000 0.995 0.998 0.770 0.548 0.792 1.000 0.987 0.523 0.999 0.987 0.525
[491] 0.792 0.530 0.999 0.996 1.000 0.990 0.999 0.997 0.972 1.000 0.532 1.000 0.000 0.000
[505] 0.997 0.999 0.507 0.803 0.511 0.517 0.509 1.000 0.996 1.000 0.000 0.796 1.000 0.000
```

```
> # Use a significance threshold of 0.05 to identify significantly different genes
> significant_genes_max = which(pvalue_all_max < 0.05)
> significant_genes_max
   [1]    7   23   35   56   71   80   85   97   98  113  121  130  138  154  157  161
  [17]  165  169  176  190  196  199  203  213  216  220  225  229  237  241  248  255
  [33]  292  298  299  300  302  303  304  307  308  323  336  342  354  357  363  386
  [49]  387  392  406  479  503  504  515  518  523  527  544  568  578  587  609  615
  [65]  620  623  625  655  659  677  683  694  710  714  721  732  734  738  769  773
  [81]  775  787  789  800  808  819  821  824  833  858  861  864  868  886  895  912
  [97]  930  946  953  959  971  984 1012 1025 1038 1073 1103 1109 1112 1124 1127 1143
 [113] 1157 1193 1205 1210 1213 1224 1243 1251 1253 1256 1257 1258 1259 1261 1262 1266
 [129] 1268 1269 1270 1271 1272 1273 1274 1279 1281 1282 1283 1286 1287 1288 1290 1291
 [145] 1293 1294 1296 1298 1301 1302 1303 1304 1305 1306 1307 1308 1310 1316 1319 1320
 [161] 1321 1323 1324 1325 1328 1329 1331 1332 1334 1336 1337 1338 1339 1341 1342 1343
 [177] 1344 1346 1347 1348 1349 1353 1355 1356 1357 1359 1361 1365 1366 1367 1369 1370
 [193] 1371 1373 1374 1375 1376 1377 1378 1383 1386 1388 1389 1391 1392 1393 1394 1395
 [209] 1396 1399 1402 1404 1406 1409 1413 1414 1416 1419 1420 1421 1423 1424 1425 1426
 [225] 1429 1434 1436 1438 1444 1446 1447 1449 1450 1454 1458 1461 1462 1465 1469 1470
 [241] 1472 1473 1474 1475 1477 1478 1479 1481 1486 1489 1493 1495 1496 1499 1500 1501
 [257] 1502 1503 1505 1508 1511 1512 1515 1516 1518 1520 1521 1525 1527 1532 1533 1534
 [273] 1536 1537 1539 1542 1544 1545 1546 1549 1551 1552 1553 1555 1565 1566 1567 1569
 [289] 1570 1572 1574 1575 1576 1577 1578 1579 1581 1583 1585 1586 1591 1592 1593 1594
 [305] 1596 1598 1599 1600 1601 1603 1604 1605 1606 1607 1608 1609 1610 1612 1613 1617
 [321] 1621 1624 1627 1628 1630 1634 1636 1637 1638 1639 1640 1641 1645 1646 1647 1648
 [337] 1650 1653 1656 1657 1661 1662 1668 1674 1677 1678 1680 1681 1683 1685 1686 1690
 [353] 1692 1693 1695 1698 1703 1705 1706 1707 1708 1710 1712 1715 1716 1720 1722 1723
 [369] 1724 1728 1729 1730 1731 1732 1733 1735 1736 1741 1743 1744 1745 1747 1750 1751
 [385] 1753 1756 1757 1758 1759 1766 1767 1768 1770 1772 1777 1778 1779 1781 1782 1783
 [401] 1784 1786 1787 1791 1793 1794 1795 1797 1798 1799 1800 1803 1804 1806 1809 1810
 [417] 1811 1813 1817 1818 1820 1821 1822 1823 1825 1829 1833 1836 1837 1839 1841 1842
 [433] 1843 1844 1849 1852 1856 1857 1858 1859 1861 1862 1865 1866 1867 1868 1869 1874
 [449] 1875 1877 1878 1879 1880 1882 1883 1884 1886 1887 1888 1889 1890 1892 1893 1897
 [465] 1898 1901 1902 1904 1905 1911 1912 1918 1920 1921 1922 1923 1927 1928 1929 1930
 [481] 1931 1932 1934 1937 1938 1942 1943 1944 1946 1947 1949 1955 1956 1957 1958 1960
 [497] 1961 1963 1965 1968 1969 1970 1972 1975 1976 1981 1983 1984 1986 1989 1991 1994
 [513] 1997 1998 1999 2000 2002 2003 2004 2006 2007 2011 2012 2014 2017 2018 2020 2021
```

Genes: 1 : 3000

H0 : max1=max2 and H1: max1−max2!= 0

p-value< 0.05 reject H0

p-value> 0.05 dont reject H0

طبق مطالبی که در قسمت میانگین گفته شده می توان خطای اول و دوم را برای **pvalue** های بدس آمده را حل کرد.

4: t.test

```
> data_P3= read.csv("Cancer_healthy_countdata.csv")
> pvalue_ttest= c()
> pvalue_ttest= c()
> for (i in 2:3000) {
+   t_test= t.test(data_P3[1:10 , i],data_P3[11:20 , i] , alternative = c("two.sided", "less",
 "greater"),
+          mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95)
+  p_value_gene_ttest=  t_test$p.value
+  pvalue_ttest= c(pvalue_ttest ,p_value_gene_ttest )
+ }
> pvalue_ttest
   [1] 3.863660e-01 5.973714e-04 5.209691e-05 3.708113e-02 1.769552e-03 2.132486e-04
   [7] 3.967567e-02 1.295368e-04 7.928482e-02 2.809052e-03 2.330171e-04 5.339766e-02
  [13] 1.568748e-03 6.216208e-04 7.410516e-04 1.954406e-05 1.650762e-02 1.927144e-01
  [19] 1.236050e-01 3.655613e-01 1.948490e-02 8.651040e-01 6.724494e-01 5.697766e-01
  [25] 1.023515e-02 1.957090e-04 2.725231e-02 2.514298e-04 1.764735e-04 2.574448e-01
  [31] 1.222038e-02 3.713253e-02 5.085551e-06 5.888577e-06 6.012073e-01 1.603579e-02
  [37] 2.132071e-03 9.359716e-04 3.267611e-05 5.037099e-04 4.114744e-04 1.789132e-06
  [43] 5.219357e-05 2.210850e-03 1.206803e-01 1.771627e-05 2.157303e-04 2.118233e-02
  [49] 7.511153e-04 2.541873e-02 5.379844e-01 2.745003e-03 4.248325e-04 1.243116e-03
  [55] 2.726064e-05 6.423799e-01 4.198445e-02 1.187630e-03 2.998794e-03 1.313377e-04
  [61] 6.949890e-04 6.792406e-01 7.989544e-04 5.817632e-02 3.352998e-02 2.422281e-02
  [67] 1.712104e-03 1.023807e-02 3.434074e-02 7.188735e-04 3.206660e-03 7.914823e-04
  [73] 8.232533e-02 7.991487e-03 2.433302e-04 1.607863e-04 5.518594e-02 1.082722e-01
  [79] 4.452881e-04 2.905453e-01 2.665448e-01 3.275863e-03 1.106001e-03 2.149776e-03
  [85] 5.434153e-01 1.610909e-03 5.640950e-03 1.793878e-01 6.444984e-07 1.037881e-02
  [91] 2.736140e-05 2.824614e-03 1.330327e-02 2.306567e-02 2.242099e-02 1.655571e-05
  [97] 4.309392e-01 4.976358e-01 2.605450e-01 6.515414e-05 1.124590e-02 1.459974e-02
 [103] 5.730910e-04 6.566822e-04 2.732562e-04 1.612204e-05 5.136909e-02 2.788757e-02
 [109] 9.036900e-01 5.596428e-02 2.056254e-06 1.312737e-05 1.257039e-01 1.325600e-01
 [115] 7.078907e-04 2.215919e-03 1.474695e-05 5.193568e-04 3.890192e-05 4.343359e-03
 [121] 5.725646e-01 6.675831e-05 8.020008e-02 8.222651e-04 4.361101e-03 3.870914e-03
 [127] 3.012087e-02 1.407496e-01 1.262535e-02 4.612916e-01 2.356569e-05 3.888531e-03
 [133] 1.892187e-02 9.351189e-02 6.918845e-03 5.030041e-02 3.531604e-02 4.111563e-01
 [139] 4.376259e-03 9.796987e-05 5.352772e-03 1.796428e-02 1.374418e-04 3.926430e-04
 [145] 1.016167e-03 6.439509e-02 1.260100e-02 2.993211e-04 6.093057e-06 1.821840e-02
 [151] 8.108914e-04 3.418398e-04 3.942194e-03 8.785741e-01 8.651267e-04 1.884847e-03
 [157] 3.768465e-01 1.079901e-01 2.360872e-04 2.377071e-02 8.451239e-02 2.228975e-04
```

```
> significant_genes_ttest = which(pvalue_ttest < 0.05)
> significant_genes_ttest
  [1]    2    3    4    5    6    7    8   10   11   13   14   15   16   17   21   25   26
 [18]   27   28   29   31   32   33   34   36   37   38   39   40   41   42   43   44   46
 [35]   47   48   49   50   52   53   54   55   57   58   59   60   61   63   65   66   67
 [52]   68   69   70   71   72   74   75   76   79   82   83   84   86   87   89   90   91
 [69]   92   93   94   95   96  100  101  102  103  104  105  106  108  111  112  115  116
 [86]  117  118  119  120  122  124  125  126  127  129  131  132  133  135  137  139  140
[103]  141  142  143  144  145  147  148  149  150  151  152  153  155  156  159  160  162
[120]  163  164  166  170  171  172  174  175  177  178  179  181  182  183  185  187  188
[137]  189  191  193  194  197  198  200  202  204  205  206  207  208  209  210  211  212
[154]  214  215  217  218  219  221  222  223  224  226  228  230  231  232  234  236  238
[171]  239  242  243  245  246  249  250  251  254  256  257  258  259  261  263  264  266
[188]  267  269  270  271  272  273  274  275  276  278  280  281  283  284  286  287  288
[205]  289  290  291  293  294  295  301  305  310  311  312  313  314  316  317  319  320
[222]  322  323  324  325  326  328  329  330  331  332  333  334  335  338  339  341  343
[239]  344  345  347  348  349  350  351  352  353  355  356  359  361  362  365  367  368
[256]  369  370  371  372  373  374  375  376  378  379  380  381  382  383  384  385  388
[273]  390  393  394  395  396  397  398  400  402  404  405  407  408  410  412  413  414
[290]  416  417  418  419  420  421  422  423  424  425  426  427  428  429  430  431  432
[307]  433  434  435  437  438  439  440  442  444  445  447  450  451  453  455  456  457
[324]  458  459  460  462  463  464  465  466  467  469  470  472  475  476  477  478  480
[341]  481  483  485  486  488  489  490  492  493  494  495  496  497  498  499  500  502
[358]  505  506  512  513  514  517  519  520  521  522  524  525  526  528  529  531  532
[375]  533  534  535  536  537  538  539  540  541  546  547  549  550  551  552  553  554
[392]  555  556  557  558  559  560  562  563  566  567  569  570  571  572  573  574  575
[409]  576  577  579  580  581  583  584  586  588  589  591  592  593  594  595  596  597
[426]  598  599  600  601  603  604  605  607  610  611  614  616  617  619  621  622  624
[443]  626  627  628  629  631  632  633  634  635  636  637  638  639  640  641  643  644
[460]  645  646  647  648  650  651  652  653  654  658  661  662  663  664  665  666  667
[477]  668  669  670  671  672  673  674  675  676  678  679  680  681  682  684  685  689
[494]  690  692  694  695  696  697  699  702  703  704  705  706  707  708  709  711  712
[511]  713  715  716  717  718  719  720  722  723  725  726  727  728  729  730  733  735
[528]  736  737  739  740  741  742  743  746  749  750  751  752  753  754  755  756  757
[545]  758  759  762  763  764  765  768  770  771  772  774  775  776  777  778  781  782
[562]  783  784  785  788  791  792  793  794  795  796  797  799  800  802  803  804  805
[579]  806  807  809  810  812  813  816  817  818  819  825  826  827  828  830  831  834
```

5: Wilcoxon test

```
######wilcoxon test

pvalue_wilcox= c()
for (i in 2:3000) {
  wilcox_test= wilcox.test(data_P3[1:10 , i],data_P3[11:20 , i] ,
  alternative = c("two.sided", "less", "greater"),mu = 0, paired = FALSE ,
              correct = TRUE, conf.int = TRUE )
  p_value_gene_wilcox=  wilcox_test$p.value
  pvalue_wilcox= c(pvalue_wilcox ,p_value_gene_wilcox )
}
pvalue_wilcox

significant_genes_wilcox = which(pvalue_wilcox < 0.05)
```

```
There were 50 or more warnings (use warnings() to see the first 50)
> pvalue_wilcox
  [1] 7.096282e-01 2.408660e-04 2.836148e-04 5.381237e-02 2.853511e-03 1.082509e-05
  [7] 2.323064e-02 1.082509e-05 1.230055e-01 1.504687e-03 7.577562e-05 2.880556e-02
 [13] 3.185649e-03 1.299011e-04 2.064143e-04 1.816511e-04 2.089242e-03 2.409676e-01
 [19] 3.766612e-01 5.451985e-01 5.830673e-02 7.325668e-01 6.719504e-01 7.053514e-01
 [25] 1.468964e-02 1.108055e-03 2.880556e-02 4.330035e-05 5.635762e-04 3.930481e-01
 [31] 1.468964e-02 2.880556e-02 2.165018e-05 1.082509e-05 1.000000e+00 6.841456e-03
 [37] 1.050034e-03 4.871290e-04 1.082509e-05 5.800604e-04 2.165018e-05 2.064143e-04
 [43] 2.165018e-05 7.252809e-04 1.931473e-01 7.577562e-05 1.082509e-05 6.301284e-02
 [49] 1.082509e-05 2.880556e-02 9.705125e-01 1.504687e-03 1.309423e-03 5.776979e-03
 [55] 1.082509e-05 9.090037e-01 3.546299e-02 7.252809e-04 5.776979e-03 1.816511e-04
 [61] 1.050034e-03 8.786000e-01 1.504687e-03 9.442541e-02 2.558130e-02 1.149624e-02
 [67] 3.597887e-03 1.082509e-05 4.885401e-02 4.871290e-04 8.055400e-03 9.054531e-04
 [73] 7.401251e-02 8.103114e-03 1.299011e-04 1.082509e-05 7.478187e-02 1.394196e-01
 [79] 4.871290e-04 2.798610e-01 3.526814e-01 3.247526e-04 1.003556e-03 1.504687e-03
 [85] 8.203966e-01 3.886207e-03 7.577562e-05 1.035892e-01 1.082509e-05 2.879473e-03
 [91] 1.082509e-05 4.540628e-03 7.087027e-03 9.004441e-03 2.329023e-02 1.806347e-04
 [97] 7.880418e-01 4.812509e-01 9.393779e-01 1.082509e-05 8.930698e-03 1.721554e-02
[103] 7.580240e-04 1.504687e-03 7.577562e-05 2.448048e-04 1.196837e-01 3.886207e-03
[109] 5.203660e-01 1.051224e-01 1.082509e-05 1.082509e-05 6.953745e-02 2.175626e-01
[115] 4.871290e-04 3.247526e-04 1.082509e-05 1.082509e-05 2.434867e-04 7.131559e-03
[121] 6.499017e-01 4.330035e-05 1.051224e-01 2.056767e-04 2.089242e-03 8.127025e-03
[127] 1.529306e-02 8.787845e-01 1.149624e-02 7.275509e-01 1.082509e-05 4.871290e-04
[133] 6.666009e-04 4.234621e-02 7.252809e-04 1.051224e-01 6.953745e-02 3.149992e-01
[139] 2.089242e-03 1.082509e-05 5.196042e-03 4.510264e-03 3.264344e-04 7.580240e-04
[145] 2.056767e-04 3.185649e-03 6.841456e-03 1.299011e-04 1.082509e-05 3.045032e-02
[151] 1.806347e-04 1.189500e-03 6.481595e-03 9.669689e-01 1.806347e-04 2.165018e-05
[157] 2.088377e-01 2.868173e-01 1.082509e-05 3.741990e-02 8.025821e-02 1.082509e-05
[163] 4.330035e-05 7.577562e-05 5.946154e-01 2.165018e-05 1.507733e-01 8.920955e-02
[169] 1.000000e+00 2.036373e-02 8.930698e-03 1.082509e-05 6.392213e-02 1.806347e-04
[175] 4.330035e-05 1.039797e-01 2.165018e-05 1.257784e-02 4.325705e-02 2.566604e-01
[181] 2.110198e-04 1.082509e-05 2.165018e-05 2.558521e-01 3.185649e-03 2.474507e-01
[187] 7.252809e-04 1.721554e-02 4.330035e-05 4.033726e-02 1.082509e-05 3.228234e-01
[193] 2.056767e-04 1.816511e-04 1.230055e-01 2.175626e-01 5.196042e-03 4.325705e-02
[199] 3.546299e-02 4.871290e-04 5.363206e-02 2.459558e-03 8.094834e-02 8.728648e-04
[205] 2.165018e-05 9.082427e-03 2.089242e-03 4.117377e-02 2.165018e-05 3.247526e-04
[211] 1.050034e-03 1.437848e-02 4.271815e-01 6.441481e-03 5.242590e-02 1.936970e-02
[217] 3.247526e-04 7.577562e-05 2.193764e-03 2.563022e-01 5.196042e-03 3.247526e-04
```

```
> significant_genes_wilcox = which(pvalue_wilcox < 0.05)
> significant_genes_wilcox
  [1]    2    3    5    6    7    8   10   11   12   13   14   15   16   17   25   26   27
 [18]   28   29   31   32   33   34   36   37   38   39   40   41   42   43   44   46   47
 [35]   49   50   52   53   54   55   57   58   59   60   61   63   65   66   67   68   69
 [52]   70   71   72   74   75   76   79   82   83   84   86   87   89   90   91   92   93
 [69]   94   95   96  100  101  102  103  104  105  106  108  111  112  115  116  117  118
 [86]  119  120  122  124  125  126  127  129  131  132  133  134  135  139  140  141  142
[103]  143  144  145  146  147  148  149  150  151  152  153  155  156  159  160  162  163
[120]  164  166  170  171  172  174  175  177  178  179  181  182  183  185  187  188  189
[137]  190  191  193  194  197  198  199  200  202  204  205  206  207  208  209  210  211
[154]  212  214  216  217  218  219  221  222  223  224  226  228  230  231  232  234  236
[171]  239  242  243  245  246  249  250  251  254  256  257  258  259  262  263  264  266
[188]  267  269  270  271  272  273  275  276  277  278  280  281  283  284  286  287  288
[205]  289  290  291  293  294  301  305  310  311  312  313  314  316  317  318  319  320
[222]  322  323  324  325  326  328  329  330  331  332  333  334  335  338  339  341  343
[239]  344  345  348  349  351  352  353  355  356  359  361  362  365  367  368  370  371
[256]  372  373  374  375  376  378  379  380  381  383  384  385  388  390  393  394  395
[273]  396  397  398  400  402  404  405  407  408  410  412  413  414  416  417  418  419
[290]  420  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435  437
[307]  438  439  440  442  445  450  451  453  455  456  457  458  459  460  462  463  464
[324]  465  466  467  469  470  472  475  476  477  478  480  481  483  485  486  488  489
[341]  490  492  493  494  495  496  497  498  499  500  502  503  505  506  509  512  513
[358]  514  517  519  520  521  522  524  525  526  528  529  531  532  533  534  535  536
[375]  537  538  539  540  541  546  547  549  550  551  552  553  554  555  556  557  558
[392]  559  560  562  563  566  567  569  570  571  572  573  574  575  576  577  579  580
[409]  581  583  584  586  588  591  592  593  594  595  596  597  598  599  600  601  603
[426]  604  605  607  608  610  611  613  614  616  617  619  621  622  624  626  627  628
[443]  629  630  631  632  633  634  635  636  637  638  639  640  641  643  644  645  646
[460]  647  648  650  651  652  653  654  658  660  661  662  664  665  666  667  668  669
[477]  670  671  672  673  674  675  676  678  679  680  681  682  684  685  686  689  690
[494]  692  695  696  697  699  702  703  704  705  706  707  708  709  711  712  713  715
[511]  716  717  718  719  720  722  723  725  726  727  728  729  730  733  735  736  737
[528]  739  740  741  742  743  746  749  750  751  752  753  754  755  756  757  758  759
[545]  762  763  764  765  766  768  770  771  772  774  775  776  777  778  779  781  782
[562]  783  784  785  788  791  792  793  794  795  796  797  799  800  802  803  804  805
```